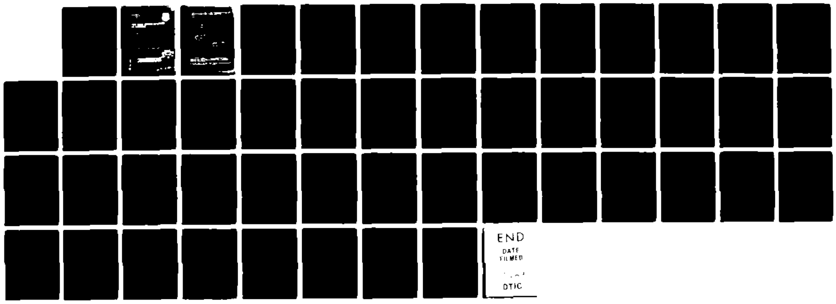
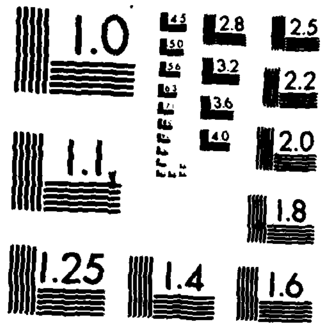
AD-A129 267 DIRECT DIGITAL TARGETING APPLICATIONS AND EXPERIMENTS 1/1
(U) ROME RESEARCH CORP NEW HARTFORD NY M J GILLOTTE
JAN 83 RRC-82-8 RADC-TR-83-14 F30602-80-C-0283

UNCLASSIFIED F/G 9/2 NL



END
DATE
FILMED
DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-14	2. GOVT ACCESSION NO. AD-A129	3. RECIPIENT'S CATALOG NUMBER 267
4. TITLE (and Subtitle) DIRECT DIGITAL TARGETING APPLICATIONS AND EXPERIMENTS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 27 Aug 80 - 30 Jun 82
		6. PERFORMING ORG. REPORT NUMBER RRC 82-8
7. AUTHOR(s) Dr. Michale J. Gillotte, Jr.		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0283
9. PERFORMING ORGANIZATION NAME AND ADDRESS Rome Research Corporation Seneca Plaza, Route 5 New Hartford NY 13413		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941723
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRRA) Griffiss AFB NY 13441		12. REPORT DATE January 1983
		13. NUMBER OF PAGES 54
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: James Sieffert (IRRA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Photogrammetry
Softcopy Mensuration
Point Transfer
Large Image Roaming
Split Screen

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document represents the final technical report for Contract F30602-80-C-0283 to the Rome Air Development Center (RADC) to provide the Experimental Photogrammetric Facility (EPF) with an upgraded Digital Image Target Location demonstration capability. The Softcopy Mensuration Software was modified to provide monoscopic mensuration capability. The DeAnza IP-8500 Display System was purchased and added to the EPF. The architecture of the IP-8500 permits large image roaming which is necessary to perform (over)

(UNCLASSIFIED)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) point transfer experiments with large digital imagery.

The software was designed in a modular fashion to permit ease in expanding the system to include multiple sensor models or even multiple implementations of the same sensor models. With this design, experiments could be conducted to compare the output object space coordinates using "improved" and "unimproved" sensor position information. Display software is independent of the sensor model and the image warping processes are designed to be a separate module independent of the sensor.

Large image roaming with image zooming was implemented as a baseline module which permits viewing any 512 by 512 pixel window of an arbitrarily large image under the control of the trackball. The design includes the use of a display status file so that the software is aware of the display register settings between modules, thus, the display will maintain the same view when entering a new (large image roaming) display module as was viewed when exiting a previous (large image roaming) display module.

TABLE OF CONTENTS

1. INTRODUCTION

This document represents the Final Technical Report for Contract F30602-80-C-0283 to the Rome Air Development Center (RADC) to provide the Experimental Photogrammetric Facility (EPF) with an upgraded Digital Image Target Location demonstration capability. This upgrade includes the modification of the Softcopy Mensuration Software (originally implemented under another contract by another contractor) to provide monoscopic mensuration capability. To further enhance the point transfer requirements, a new state-of-the-art display configuration was purchased and installed in the EPF. This display, the DeAnza IP-8500 Display System, is designed to permit the expansion of the display memory to 4 megabytes for display purposes and and additional 1 megabytes for use as "scratch-pads". The IP-8500, with its split screen mode permits large image roaming which architecture of the previous display system on the EPF (COMTAL 8300) could not support.

The software was designed in a modular fashion to permit ease in expanding the system to include multiple sensor models or even multiple implementations of the same sensor models. With this design, experiments could be conducted to compare the output object space coordinates using "improved" and "unimproved" sensor position information. Display software is independent of the sensor model and the image warping processes are designed to be a separate module independent of the sensor (although the warping parameters could be derived using the sensor characterists).

Large image roaming with image zooming was implemented as a baseline module which permits viewing any 512 by 512 pixel window of an arbitrarily large image under the control of the trackball. This large image roaming capability was implemented so that the image could be loaded using operator specified coordinates thus permitting that the initial view window have any arbitrary (operator specified) coordinate as its upper left corner. The design also includes the use of a display status file so that the software is aware of the display register settings between modules, thus, the display will maintain the same view when entering a new (large image roaming) display module as was viewed when exiting a previous (large image roaming) display module.

The operating system used for the implementation of the DDT system is the Programmers' Workbench version of UNIX (PWB/UNIX) as provided by Bell Laboratories. To enhance its capabilities, software developed by the University of California at Berkley was integrated into the PWB/UNIX. UNIX was used as the operating system because of the display and mensuration software which was implemented under UNIX by the AFES contract. This operating system, in addition to the use of existing application software, was also modular in a fashion which makes it easier to conduct experiments using portions of the application software, or to use the application software as a complete unit.

The use of PWB/UNIX also made it easier to control the software development during this contract and will make it easier for RADC to maintain the software after the completion of this contract. The software developed during this contract includes administrative tools which include the Source Code Control System (SCCS) and the Make utility. These two intricate tools are integrated into the DDT system administrative control structure.

In addition to the photogrammeteric application programs, the DDT system also includes software pertaining to image compression and decompression as required by the statement of work.

1.1 REPORT ORGANIZATION

- Section 2 in this document discusses the operating system PWB/UNIX in greater detail than this introduction.

- Section 3 discusses the DeAnza IP-8500 Display System.
- Section 4 discusses the application programs implemented and integrated into the DDT system.
- Section 5 provides an overview of the Administrative Controls of the DDT system.
- Section 6 consists of conclusions pertaining to the accomplishments of this contract and also provides recommendations of ways to improve and complete the DDT system.

2. UNIX SYSTEM OVERVIEW

The DDT system design builds on the UNIX operating system. UNIX provides an environment in which program development and system integration is extremely straightforward. It strongly encourages and extensively supports modular development. These are the requirements of the DDT system in which flexibility without sacrificing ease of use is one of the most important criteria.

A wide variety of very powerful and flexible programming tools have also been developed for use under UNIX. These tools make systems very easy to manage and control. The UNIX operating system itself and a few of the more important programming tools are described in the following paragraphs.

2.1 BACKGROUND OF THE UNIX OPERATING SYSTEM

The UNIX operating system dates back to 1969 when Ken Thompson of Bell Laboratories began creating it on a PDP-7 minicomputer. His primary goal and that of the others that soon joined him in his work was to create an environment in which they could comfortably and effectively pursue their programming research work. UNIX therefore became unique in that the major emphasis was not given solely to running programs but to creating them. Most operating systems do a good job of efficiently running programs but present many obstacles to efficient programming. Insufficient effort is usually made when designing and building the operating system to provide a friendly environment for the programmer. This is of particular importance to the programmer whose prime motivation is not in programming per se but in using programming as a tool in research and development in a variety of application areas. The authors of UNIX placed special emphasis on the needs of the programmer and as a result have created a very sophisticated operating system that is surprising in its simplicity and yet very general in application. These characteristics make it extremely intelligible and easy to use.

The basic philosophy of UNIX is to provide the necessary operating system functions consisting of a file system and an environment in which programs can be developed and executed. It was the firm belief of the authors that an operating system should not be concerned with providing functions that are specific to an application or task. It should provide the basic framework through which the hardware is accessible but not obscurred with unnecessary application specific capabilities. The more specific capabilities were believed to belong in the user level software where they would not burden all programmers and programs. The result is a very sophisticated but simple and intelligable operating system.

2.2 THE C PROGRAMMING LANGUAGE

Along with the UNIX operating system the C language was developed which is based in large part on the BCPL language developed by Martin Richards. C is the main programming language in UNIX although most other major languages are available including LISP which is commonly used in the Artificial Intelligence community. In fact the UNIX system itself is written entirely in C with the exception of two machine dependent modules. This makes UNIX a very portable system which is evidenced by the fact that UNIX has been implemented on PDP-7, -9, -11/20, 11/34, -11/40, -11/60, -11/70, VAX-11/780, and Interdata 8/32 computers. Recent efforts have been completed to implement it on several microprocessors namely the Intel 8086, the Zilog Z8000 and the Motorola 68000. Therefore C makes UNIX a truly portable operating system which is not made obsolete by new generations of hardware.

The C language itself is a structured language with modern control flow and data structure capabilities. It is both a low and a high level language in that it provides access to basic machine level operations, yet it is easy to use and is totally general in application. It is not difficult to learn and, if written properly, is easy to read. C compilers now exist for a variety of machines and operating systems including the Honeywell 6000, the IBM System/370, and the Interdata 8/32 as well as the DEC 11 line of computers.

2.3 THE SHELL - THE UNIX/USER INTERFACE

The standard interface between the programmer/user and the operating system is a command interpreter called the shell. The shell interprets user command input and takes the appropriate action. The philosophy of the shell stresses simplicity. Most commands are implemented in entirely separate and independent processes. The interface between the shell and the commands is very straightforward in that it is comprised mainly of character strings which consist of the command and any arguments accompanying it. Special wild card operations such as all file names containing a specific character pattern or patterns are performed by the shell so that consistent results are obtained system wide.

Open files can also be passed to a process by the shell (or by any process so written) which provides a tremendous degree of flexibility in terms of combining several basic functional processes to perform a more complex operation. Processes combined in this manner are typically (although not necessarily) combined in a pipeline fashion where the output of one process becomes the input to another. A concept of standard input and standard output exists for each process by which this is accomplished. Standard input and output are simply I/O channels referenced by specific file identifies called file descriptors similar to logical units in FORTRAN. By default these channels are the terminal from which the command was given. However, the shell can assign any open file to these channels. An open file can be a pipe which can be either read or written. By passing the appropriate ends of pipes as open files corresponding to the standard inputs and outputs of the processes, a pipeline is thus formed. Therefore, each process in the pipeline becomes a filter that accepts data from its input, processes it in some manner, and writes the result to its output which in turn is the input to the next process in the pipeline.

The processes themselves are not aware of the source of their input or the destination of their output. The sources and destinations can be files, pipes or devices (including terminals). In fact the UNIX file system treats files and devices alike in that the same naming syntax is used in forming the identifiers called pathnames. This provides a totally general and straightforward mechanism for accessing system resources.

Since the shell can cause processes to be executed either singularly or in combination, it logically follows that one can program the shell itself to perform desired operations. To enhance this capability the shell is provided with control-flow primitives and string-valued variables. The result is a very high level programming capability which allows very complex operations to be performed with very brief shell programs or scripts. A typical example of this is the Automatic Feature Extraction System (AFES) developed by PAR under Contract F30602-78-C-0080 which draws heavily on the shell's capabilities. That system is made up of a variety of basic processes consisting of the standard UNIX utilities as well as application specific modules written at PAR, all tied together by a variety of shell programs. The result is a very modular, flexible, and powerful system that is easy to use and manage.

The DDT system will rely upon the shell for its modularity, flexibility, and power in much the same manner as the AFES has done. In fact, considerable software will be drawn from AFES in building the DDT system. DDT will consist of a variety of modules (commands) that will each perform a basic function. A set of shell programs will also be written which will combine certain of the basic commands to form higher level commands. In a similar manner the DDT user may combine commands in a shell program to design experiments as desired. The procedure for accomplishing this is simple and straightforward. No real programming experience is required to accomplish this task.

2.4 SOURCE CODE CONTROL SYSTEM AND MAKE FACILITY

Finally, UNIX has a very extensive set of text handling capabilities from controlling program source to document preparation. The programmer finds the Source Code Control System (SCCS) to be extremely valuable. It provides an orderly manner in which source code, independent of language, can be created, updated, and stored. At any time the programmer can see a complete history of a module's development as well as recover the state of the module at any stage in its development.

Closely allied with SCCS is the Make facility. This provides a logical mechanism through which the programmer can describe how a process or a complete system is constructed and the dependency relationship between the modules. After modification of any given module the programmer merely need give a single command and the Make system will insure that all processes that are affected by the change will be recompiled and/or rebuilt. Processes that are not affected by the changes(s) are not rebuilt.

The AFES used the SCCS and the Make facility to provide an environment in which an entire system can be constructed and maintained by a team of programmers under the control of one person called the system administrator. Each member of the team can modify any portion of the system without affecting any other member's work. This is accomplished by localizing the changes to the individual programmer's directory. However, the modifications can be tested within the context of the entire system, and, when ready they can be incorporated permanently into the system by the system administrator. This entire capability was incorporated into the DDT system. A more detailed discussion about this capability is contained in section 5.

3. DEANZA IP-8500 DISPLAY SYSTEM

The state-of-the art in display systems today is represented by the DeAnza IP-8500 Display System. It has a number of features which make it very well suited to the DDT application. Of particular importance are the flexible memory architecture which can be reconfigured under software control. This is especially useful in the image roaming application.

In the earlier display models, such as the IP-5500, a memory mapping interface was available which allowed direct access to individual pixels through the host processor's UNIBUS address space. This is very powerful for applications requiring random access to pixels, such as image resampling. But it is not suited for situations in which high image data I/O rates are required, such as for image roaming. The IP-8500 was chosen for the DDT system because of the need for an effective image roaming capability.

The major I/O path for the IP-8500 is through the Direct Memory Access (DMA) channel. The efficiency of direct host access to refresh memory achieved in IP-5500 was sacrificed in the interests of the added I/O speed achievable through DMA transfers. However, much of the flexibility for loading image memories was retained and enhanced and supports both programmed I/O and DMA transfers. There is direct hardware support for loading display memory of arbitrary size with image data. This feature was specifically designed to support large image roaming.

Another major capability added to the IP-8500 is the ability to handle large images. This is done by providing the capacity for up to twenty memory boards each of which can be 512 x 512 x 8 bits or 1024 x 1024 x 8 bits. Sixteen of these boards can be used as refresh memory while the remaining four are reserved for buffer operations. The high density 1024 x 1024 boards were not available at the time the DDT system was purchased; it has a total of 6 memory 512 X 512 X 8 bit memory boards. This configuration provides an effective image roaming capability based on tessellated images, accompanied by a split screen display to facilitate point positioning.

The memory boards can be configured in a variety of ways for both host I/O operations and for image display refresh. Configurations of multiple channels can be established under software control for I/O where each channel can consist of rectangular arrays of memory boards up to a maximum dimension of 2048 by 2048 (4096 by 4096 if high resolution memory boards are used).

The IP-8500 has considerable flexibility in its generation of output video from the the imagery stored in the frame buffer. Up to four Video Output Controllers (VOC's) can be configured in one display system where each VOC can drive one 512 x 512 full color monitor (or 1024 x 1024 with the high resolution option). Each monitor can have an interactive device (trackball, joystick, or tablet), an alphanumeric overlay, and a dual cursor generator associated with it. The four VOC's draw their inputs from the common pool of sixteen refresh memory boards. Assignment of the memory boards to each monitor can be done dynamically via the VOC's under software control. The flexibility provided by these capabilities allows the single display system to be configured as one multiple display workstation, several individual workstations, or some combination in between.

The VOC's also provide a split screen capability which effectively allows the refresh memory to be configured in a square array of four memory boards for output viewing. This is extremely useful for large image roaming because memory can be loaded in background to maintain refresh image data beyond the currently visible portion. As the image is scrolled, the nonvisible data is immediately available to be scrolled onto the display monitor.

The IP-8500 also supports a Digital Video Processor (DVP) which is a pipeline processor capable of absolute or two's complement arithmetic operations on 8 or 16 bit pixel data. The operations available are similar to those in the IP-5532 DVP. As with that display system, the IP-8500 DVP performs its operations on the entire input array in one refresh frame time. Typically such operations as convolution, correlation, edge detection, edge enhancement, and image merge can be done with the DVP.

A possible future addition to the DDT IP-8500 configuration is a special purpose DVP available for warping operations. That processor is capable of a second order polynomial transformation in one-fifth of a second.

4. APPLICATION SOFTWARE

The general philosophy of the software development was to generate modular software which separates the mensuration requirements into logical components. The software was broken out into the following different components:

- Display,
- Point Positioning/Mensuration,
- Digital Terrain Elevation Data,
- Image Compression/Decompression.

It is clear that the image compression and decompression algorithms should be separate from the other aspects of the application software, simply from the standpoint that the hardware configuration of the Experimental Photogrammetric Facility does not lend itself to a rapid throughput rate. A rapid throughput rate would be desirable to permit viewing an image that is stored in compacted format.

The separation of the display software from the photogrammetric software is a rational approach, because the viewing of an image does not normally require the interior or exterior orientation parameters. Also, if the two were not separate, then the display software would have to be constantly expanded every time a different sensor mathematical model is to be added to the system. In that case, either the display software modules would continue to grow in size, because of the inclusion of the additional code, or the display code would have to be duplicated for each different sensor system. In either case, this causes software reliability problems for the display code. The inclusion of additional code could cause the previously working display code to start failing, or improved display functions would need to be inserted in many different programs.

Likewise, the photogrammetric operations would be inconvenient if the derivation of ground coordinate required one to display an image and select the display coordinates of the desired target. This would introduce random error from operator input, even when the image coordinates are already known. Thus, the display and photogrammetric modules were implemented in separate modules so that the coordinates derived from the display could then be passed to the photogrammetric routines, while other independent methods of deriving the image coordinates could also be passed to these modules.

The decision to separate the digital terrain elevation data from the other software was made because RADC had another contract in which terrain intersection techniques were being investigated. The DTED operations were separated from the photogrammetric operations, so that the inclusion of various or different terrain intersection algorithms would be a simplier process.

4.1 DISPLAY SOFTWARE

The development of the display software was hindered by various factors discussed in Section 3.0 concerning the DeAnza IP-8500. It was expected that the software developed under the AFES contracts would be readily usable for the DDT system. Thus, the major software effort was directed to achieving the capability of <u>large image roaming</u> on the IP-8500 Display System. With the dynamic nature of the display, and the mensuration requirement, it was essential that the <u>status of the display</u> be stored. This provided a "<u>re-entry</u>" <u>capability</u> of the "roaming" programs and provided sufficient image size and display parameters that the coordinates of an image object could be computed regardless of the scale and position on the display. Each of these items are discussed in more detail in the subsections that follow.

As an aside, before providing the additional discussion in the subsections, two other issues should be mentioned: the COMTAL display and the AFES display software.

- The COMTAL was planned to be used as an overview display, i.e. a view of an image at a reduced resolution, thus providing a larger viewing area. The IPS had a COMTAL driver for the RADC IPS configuration; this driver was to be installed on the EPS within the EPF. For some reason, the driver did not work directly on the EPF's COMTAL, thus the driver was modified. Before the driver was completed, the hardware had a severe problem. The IPS COMTAL was then moved to the EPF. The last attempt to use the modified driver on that COMTAL display failed. It is not clear whether the failure was due to a difference in the two different COMTAL displays, or whether it was due to a software error. All display software associated with the COMTAL was never tested.

- The AFES display software was to be incorporated into the EPF as much as possible. The major items to be included was the image enhancement modules. The usefulness of the enhancements is that an image interpreter could modify the image intensities which may improve the coordinate selection process. Unfortunately, the DeAnza IP-8500 was not upward compatible with the previous models contained on the AFES configuration. Furthermore, as a result of this incompatibility, no image enhancement programs were implemented on the IP-8500.

4.1.1 Image Roaming

The need for large image roaming is apparent when one considers the general mode of operation that an image interpreter uses to locate a portion of the scene when using hardcopy imagery. Typically, the interpreter using hardcopy is able to view a substantial portion of the image, thus having available cues to locate the desired subscene. With softcopy viewing, it is convenient from a programmer's viewpoint to limit the viewing area to that of the display screen dimensions or the display's memory dimensions. This places an arbitrary limit on the scene area available to the image interpreter. A great deal of effort was expended to achieve the large image roaming capability.

The method of achieving this capability involved optimized image storage techniques (blocking the image in "tessellations" and storing the imagery in a "nearly contiguous" fashion) and taking advantage of the display capability of the IP-8500 Display System. The importance of the optimized image storage lies in the "timeliness" of the image roaming. It is very annoying to be roaming an image only to have the roaming temporarily suspended while the display updates its image contents. If the image data are not stored in tessellated format, then roaming the image a single pixel interval in the left or right direction would require a disk read for each of the rows of the image stored in the display memory (just to update a single column). If the display memory corresponds to the display viewing area, then 512 disk sector reads would be required to obtain the 512 bytes to update the display. Even with the storage of the image in tessellated format, the updating of each tessellation requires the accessing of 16K bytes (128 pixels by 128 pixels) from disk storage. Using the normal storage technique of the PWB/UNIX system would require 32 separate system calls to read the single tessellation from the disk. Because this would slow down the image roaming, a high speed file access software package was developed. With this software package and contiguous allocation, the 16K bytes of data have been accessed with 1-5 system calls instead of 32, a substantial savings.

4.1.1.1 Tessellated Image Storage

The image format has been defined to store the image data in sub-images or tessellations. This means that instead of storing the image in scan line format (consisting of a stream of bytes for a single scan line, followed by another stream of bytes for the next scan line, etc.), the image is partitioned into sub-images, each of dimension "n"x"n", and each sub-image is stored in scan line format, followed by the next subimage.

```
*****************************************************************
*       *       *       *       *       *       *       *       *
*   1   *   2   *   3   *  ...  *       *       *       *   m   *
*       *       *       *       *       *       *       *       *
*       *       *       *       *       *       *       *       *
*****************************************************************
*       *       *       *       *       *       *       *       *
* m + 1 * m + 2 * m + 3 *  ...  *       *       *       *       *
*       *       *       *       *       *       *       *       *
*       *       *       *       *       *       *       *       *
*****************************************************************
*   .   *       *       *       *       *       *       *       *
*   .   *       *       *       *       *       *       *       *
*   .   *       *       *       *       *       *       *       *
*       *       *       *       *       *       *       *       *
*****************************************************************
*       *       *       *       *       *       *       *       *
*(p-1)m *  ...  *       *       *       *       *       * p x m *
*  + 1  *       *       *       *       *       *       *       *
*       *       *       *       *       *       *       *       *
*****************************************************************
```

Figure 4-1 Illustration of the Tessellated Image Format

In Figure 4-1, each rectangle corresponds to one subimage or one tessellation. The order of the image data stored in the file is subimage 1 stored in scan line format followed by subimage 2, subimage 3, ..., subimage~m subimage~m+1, and so on. The dimension of the tessellations used on the DDT system was 128 pixels by 128 pixels. If an image is stored, this format is roamed one pixel to the left or right, a single disk sector read updates 4 pixels along the side of the image. This is an improvement over the 512 disk reads cited above, because the 512 pixels could be obtained by 128 disk accesses. This by itself is not, however, adequate to achieve reasonable image roaming capability. It would appear that smaller tessellation dimensions would be more appropriate. For example, use of 64~x~64 tessellations would further reduce the disk accesses by a factor of two. The decision to use a larger value was premised by the capability of the IP-8500 Display System, will be discussed in a subsequent subsection. In fact, 128~x~128 is the largest dimension that could be used and still achieve large image roaming (cleanly) while viewing 2 different images in the left/right or top/bottom display mode.

The final design allows the system to update 128~x~1024 pixels or 1024~x~128 pixels using approximately 8 disk accesses and 8 display memory accesses. This is only possible through use of the unique combination of the tessellation format of the image data, the high speed file routines and the architecture of the display system.

4.1.1.2 High Speed File System

The High Speed File Routine (HSFR) package is a set of user subroutines that allow very efficient reading and writing of large files. These routines permit application programs to achieve raw disk speed data transfers rates while still maintaining a UNIX file system structure. The HSFR's use the UNIX file system data structures, but simply rearrange the data blocks in the desired order for efficient large data transfers. The HSFR's take advantage of contiguous file allocation which permits the software and hardware overhead of a file access to be spread over a large number of disk sectors. It also relieves the operating system from having to buffer and copy all transferred data. The improvement over the standard file system is substantial. Timing results included at the end of this subsection verify this assertion.

UNIX file systems typically allocate files in an interleaved fashion. This allows several single blocks to be processed in a single disk revolution. In fact, a UNIX file system is usually set up in such a way that blocks are allocated at physical address intervals where rotation latency between blocks coincides with the software overhead of a single block access. The UNIX file system is efficient for small files; however, for certain special applications (e.g. image processing) in which it is known that large files are to be accessed, the normal UNIX file system is inefficient. Instead of using the standard interleaved method of generating the free block list, the HSFR system assumes that the list was generated in numerical order.

The HSFR package is designed to work correctly (although at reduced transfer rates) even when the files being read or written are not 100% contiguous. This can be helpful when it is awkward to allow c_creat to take the file system off-line. The normal system file copy utilities or existing image creation programs may be used and reasonable performance is expected. A word of caution is in order: it is important that the file system be used only for large files; adding and deleting small files will fragment the ordering of the free block list and produce fragmented files.

One current problem is that if a file from a standard file system is used with HSFR, there may be almost no contiguous blocks in it. The program memory allocation for the physical blocks would have to be enlarged to handle all of the blocks in the file. An alternative to enlarging the buffer is to detect the buffer overflow and set a flag which indicates that the file should be opened and accessed using the standard "open", "lseek", "read" and "write" utilities. Both solutions, although degrading the performance of the data accessing, and hence the large image roaming for images inappropriately stored, would permit large image roaming for those cases where it was impossible for the user to allocate sufficient contiguous storage for the image. Neither solution was implemented.

The data presented in Table 4-1 applies to a PDP 11/45 system with an RP06 disk. The times are for transferring 1 Mbyte file into memory. There are 2048 sectors in a 1 Mbyte file. An RP06 has 22 sectors per track and 19 tracks per cylinder. One disk rotation can yield at most 22 sectors. A disk seek is required after 418 sectors of a cylinder have been read.

Table 4-1 Disk Access Performance Timings

The theoretical best possible transfer rate with perfectly placed disk blocks (93 revs + 5 seeks)	1.6 sec (calculated)
512 byte block, with interleave = 9 (19x9x5 revs + 5 seeks)	14.2 sec (calculated)
Actual read performed on PWB/UNIX system, 512 byte block, with interleave = 9	15 sec (measured)
Contiguous file system, 8K byte transferred/revolution (128 revs + 5 seeks)	2.2 sec (calculated)
Actual read performed on PWB/UNIX system, 512 byte block, with contiguous file system, 8K byte transferred/revolution (128 revs + 5 seeks)	<3 sec (measured)

The table illustrates that the HSFR performs close to the theoretical best possible transfer rate. The use of HSFR greatly improved the effectiveness of the large image roaming.

4.1.1.3 DeAnza IP-8500 Split Screen Capability

The DeAnza IP-8500 has a split screen capability which permits partitioning the video display into 4 parts (controlled by a horizontal and vertical split). For each partition the display may derive the video signal from up to 4 distinct image memory boards or channels. Since each board can be "zoomed" and "scrolled" independently of the others, an image of dimension

1024~x~1024 could be loaded into 4 memory channels and the display used to provide a view "window" having dimension of 512~x~512. In this manner, it is possible to perform image roaming and zooming on any subimage of dimension 1024~x~1024 loaded into 4 memory channels. Image wrap-around will occur unless the memory boards are scrolled with the "zero wrap around" bit enabled.

Although roaming/zooming within a 1024~x~1024 subimage is attractive, the restriction of limiting the image interpreter's view to this subset of a large image is what was referred to earlier in this report as a "programmer's" convenience. The DeAnza hardware design permits "matrix mode" writes, as opposed to the previous standard of many display systems to write a complete scan line for every write. In matrix mode one specifies a starting x,y memory board address and matrix dimensions. The byte stream written to the display memory will modify only image data within that matrix having the starting x,y coordinates as one of the corners of the matrix. By using this "random access" addressing of the DeAnza memory, and the "wrap-around" characteristic of the image roaming, a complete large image stored in tessellation format may be viewed using image roaming. To illustrate the technique, as the image is "roamed" to the right (i.e. the viewing window moves to the right), the image wraps-around on the left side of the display. But as the window moves to the right, image data associated with the tessellations to the left is then read from the disk and written to the display memory in a manner such that this updated scene information is the data that is "wrapped-around". Thus, the module that performs the large image roaming manages the display registers associated with the trackball and trackball switches, the cursor, split screen registers, zoom/scroll registers, and the split screen video assignments. It must also maintain the status of the image and the display to determine what portion of the image must be used to update the display memory as the image is roamed. Finally, it must read the image data in a timely fashion. This is accomplished by storing the data in tessellated format and accessing the data using the high speed file routines.

4.1.2 Split Screen With Roaming

The use of split screen viewing along with roaming is desirable for the point transfer task. Point transfer is the first step in computation of ground coordinates of targets when the available sensor collection parameters are not adequate to derive ground coordinates directly. Viewing the recce imagery while also viewing a data base image facilitates point transfer.

Two programs were implemented which provide the capability to view two separate images at the same time in a "split screen" mode. In the first case, the program has the options to use top/bottom or left/right split screen mode, while only one of the images has the large image roaming enabled and the other image remains fixed. In the second case, the program is designed to display the images in top/bottom split screen mode while having the large roaming capability enabled alternatively between the two portions of the display. When one image has roaming enabled, the other image remains fixed. The technique for performing large image roaming while in the split screen mode is similar to that for the full screen mode except that only two image memory channels can be allocated to the roamed image. This is the case because the "split" is always fixed (at the middle of the display), thus the image can be roamed only in the single memory board in the short dimension.

The design of the display makes it impossible to roam on both sides of the split on the display with the roaming occurring in the horizontal direction (in the case of top/bottom display mode) at separate rates. In the case where the software alternates between the top and bottom large image roaming, the spare memory boards are used to create a "fixed" image which corresponds to the image that is not to be roamed. In this manner, the scene may be viewed with no apparent change in context while switching to enable roaming on the other image.

It is this program that should be used as a starting point to design and implement suitable point transfer programs, whether the technique be as simple as the "eye-ball" transfer technique or the more complicated analytical point transfer technique. In the latter case, the ability to use two cursors, one slaved to the other via a transformation might be desirable.

4.1.3 Display Status File

The display status file has two major roles in the use of the IP-8500 for large image roaming.

- The first role is to serve as the storage medium for the "interior orientation" information which permits the conversion of display coordinates to image coordinates.

- The second role is to store the image scroll and zoom register values for the memory channels. These values must be stored because the DeAnza IP-8500 does not permit the interrogation (reading) of those registers as in the case of nearly all of the other display registers.

The display status file is loaded with parameters when images are loaded into the display image channels. These parameters are used to assure cohesive viewing when the images are manipulated by the image roaming programs. They also provide image coordinate information which is used to manage the image updating portion of the large image roaming task, as well as providing subimage locations for the display to image coordinate transformation.

4.1.4 Re-entry

Another feature that was included in the design of the display status file is to permit the re-entry of any of the image roaming programs while still preserving the current viewing scene. It is very restrictive and distracting to roam through an image locating a certain window of interest, only to exit the program (accidently or on purpose) and upon re-entry (or initiation of another image roaming program) have the display provide a view of another window. This can cause the analyst to lose context of the scene while he is trying to relocate the original window of interest.

4.2 POINT POSITIONING/MENSURATION SOFTWARE

The principle function of software developed during the DDT effort was to perform image to ground coordinate computations. This capability also exists within the EPF on the PTS-STM hardware/software system. The difference between that system and the DDT system is that the former uses hardcopy imagery, while the DDT system uses softcopy imagery. Essentially, on the PTS-STM system, the analyst "floats the dot" and the system derives the ground coordinates using the interior orientation, exterior orientation and the specific mathematical representation of the sensor geometry. The analyst has the option of changing the scale and orientation of the view by using the optics. The software uses parameters which account for these effects in the conversion of stage coordinates to photo coordinates.

The basic operation for the softcopy point positioning/mensuration should be basically the same. The distinction is that optics are not available to perform changes in orientation or scale. These must be performed digitally, i.e. the image must be "warped" to change the scale or to orient the image to permit viewing a stereo pair.

The software developed for the DDT system was designed in a modular fashion to permit much flexibility for future expansion. The basic starting point or hypothesis is that the image-to-ground coordinate computation can be separated into three parts, with the first two independent of the sensor used to collect the image. These parts include:

- Display to Image Coordinates,
- Image Coordinates to Sensor Coordinates, and
- Sensor Coordinates to Ground Coordinates.

The first of the three is concerned with the selection of a point on the display, which must be transformed to a specific image coordinate within the image file's coordinate system. This operation does not require any knowledge about the sensor system or even the interior orientation parameters. Thus, because of its relationship to the display, the display-to-image transformation was included as part of the display software. The second operation is essentially an implementation of an interior orientation transformation. Again, this operation, aside from different transformation techniques, can be thought of as being independent of the sensor system. Calibrated sensor coordinates are used to derive the transformation parameters, but, once the parameters are derived, the application of the interior orientation does not require knowledge of the sensor. It must take into consideration any rotations or scale changes that were performed to "bring the pair of images into stereo". Again, this may be based on the sensor collection properties, or it could rely on a simple trial and error procedure performed by the operator. Regardless of the technique used to generate these transformations, the process of computing the sensor coordinates using the image coordinates is still based on the application of the warping parameters. Finally, the third operation does require sensor specific information (contained in "photo headers" within the DDT system) and that information can be stored in a file corresponding to that particular softcopy image. Different sensor types require different photo header formats and different algorithms to perform the computation from sensor coordinates to ground coordinates.

4.2.1 Softcopy Mensuration

The Softcopy Mensuration software was written in FORTRAN for a VAX Computer with a VMS operating system by another contractor for the SCM contract. Additional software was available which was also developed under the SCM contract for RADC by another contractor. This software was interfaced to the Digital Image Processing System (DIPS) on the EPF, but after examination, was not considered pertinent to the DDT A&E effort. The VAX SCM software which was used during the DDT A&E contract consisted of 2 major modules. The first created test sensor parameters which were assumed to be used to establish control over the input parameters, thus permitting the output to be examined for accuracy. The second module, which was the major component of the SCM software, it performed the following tasks:

1. "Pre-digestion" of the sensor exterior orientation parameter for a pair of images.

2. Select an approximate image coordinate and compute an approximate ground coordinate by intersecting with the ellipsoidal model of the earth.

3. Compute warping parameters for the pair of images in a neighborhood of the selected point.

4. Perform the two image warpings that would permit the images to be viewed in stereo.

5. Display the warped images in anaglyph mode.

6. Interact with the display to "float-the-dot."

7. Project the image points to the ground and iterate to a ground coordinate.

The following observations were made.

- First, the "pre-digestion " of the sensor exterior orientation parameter was a time consuming operation and once it was performed there was no reason to perform the operation again. Also, the software computed the parameters for the two images separately, although the results were placed in a single data file.

- The computation of the warping parameters need not be re-computed every time the program is executed. The parameters could be stored in a file and the images could also be warped and saved for future processing. Even though virtual addressing on the VAX makes the warping program easier to perform, there is still no reason to force the warping to be performed every time the program is executed.

- The display aspects, as indicated in previous sections of the report, are really independent of the sensor geometry, and hence can (and should) be made a separate program.

Since the computer configuration that was used to modify the SCM software was a VAX with a UNIX operating system, the FORTRAN code required modification to be compatible with the f77 compiler. In an attempt to accelerate the modification cycle, the SCM was partitioned into more modular software. The following modules were extracted/developed:

1. Creation of test exterior orientation parameters for a pair of images, but stored in separate "photo headers."

2. First phase of pre-digestion of the exterior orientation parameters for a single image.

3. Edit capability for the above parameters. (This permits experimentation to examine the accuracy trade-offs by modifying the exterior orientation parameters).

4. Second phase of pre-digestion of the exterior orientation parameters.

5. The above edit capability could also be used at this point too.

6. Computation of image warping parameters for a pair of images. (Note that only parameters are output - no warping is performed.)

7. Approximate computational ground coordinate from selected image coordinate. This was modified to allow intersection with any elevation above the ellipsoid, rather than restricting it to zero elevation above the ellipsoid. (This serves as the "mono scopic" technique using a "nominal" earth model.

8. Computational ground coordinates assuming that a pair of conjugate sensor coordinates are known. This was performed two different ways:

   - Duplicating the SCM software received. This method had obvious software bugs in the original SCM code.

   - Performing the standard "ray-intersection" technique.

Note that it was not necessary to write special software for the imagery associated with the SCM software to perform the image warp, displaying the SCM imagery and "floating-the-dot". These were to be developed for general imagery and not be restricted to the SCM imagery.

The results of the SCM software modifications were tested in a number of ways. Unfortunately, it was impossible to certify that the results were accurate. The original SCM source software had a program which generated a test set of parameters for the exterior orientation parameters. The source file also had comments which defined what the ground coordinates should be for certain sets of image coordinates. Both SCM computed ground coordinates and the "surveyed" ground coordinates were included in the source file. By using these test constants, computed ground coordinates for specific image coordinates could be compared with the answers provided in the source file. Unfortunately, the modified software never generated answers equal to these included in the comments, but at the same time, it was impossible for the original software to be tested to verify that the computed coordinates agreed with these in the source comments of the test EO generator. (Keep in mind that the SCM software had obvious bugs, thus it is unclear whether the original SCM software was actually generated the "computed" results reported in the test program.)

The status of this software is that it was successfully demonstrated on the VAX UNIX system, but has not yet been transferred to the EPF PWB/UNIX operating system on the PDP 11/45.

4.2.2 AFES Mensuration

The mensuration software of the AFES provided a basic starting point for the design of the DDT system for the EPF. The AFES already had mensuration software, including image warping software to permit stereo viewing with proper image orientation, for the frame geometry sensor. In addition to performing the image coordinate to ground coordinate computations, the software for stereo viewing, stereo point selection, and general ground coordinate transformations were already available on the AFES. The file structure convention used by AFES permitted maintaining the "history" of interior orientations so display coordinates could readily be transformed back to the sensor coordinate system. "History" of interior orientations refers to the transformations used to map an image coordinate of a "daughter" image (subimage, warped image, rotated image, etc.) to its corresponding position in its "mother" image (which could be the original image or even a subimage, warped image, etc. as well).

Certain limitations of the AFES were imposed by the design of its mensuration software. In particular, the software design required "floating-the-dot" every time a ground coordinate was desired. Likewise, there was no facility available which permitted simply extracting image or sensor coordinates. If sensor coordinates were known, there was no way to project the position to the ground coordinate system. The design of the "photo header" was limited to that of frame geometry, thus limiting the expansion to other sensors.

It was essential that the software available from AFES be modified so that it met the system objectives of the DDT A&E effort. The DDT effort required a system which could experiment with different types of sensor geometric mathematical models, thus providing a method to access the accuracy of various mathematical models and study trade-offs between processing time and coordinate accuracy represented by simpler algorithms. To meet these goals the AFES mensuration software was modularized in a manner similar to that applied to the SCM system. The results:

- The display software was modified to compute image coordinates, using a "pseudo-interior orientation" transformation, of points selected on a displayed image.

- Software was implemented to transform image coordinates to sensor coordinates using the sequence of "interior orientations" from daughter image to mother image. This is accomplished by using the transformation type and parameters used to generate the daughter image from the mother image.

- A sensor-to-ground transformation module, was extracted from AFES, which uses image coordinates captured in a stereoscopic fashion. This uses the standard "ray intersection" method. The photo header is accessed via the header name, instead of always requiring the name to be "phdr.l" and "phdr.r". In this manner, various versions of a photo header could be used instead of "phdr.l", thus permitting the use and comparison of different models.

- Another sensor-to-ground transformation module, was also extracted from the AFES software, which uses image coordinates captured in monoscopic fashion. This module intersects the ray with the local vertical coordinate system (with the origin stored in the photo header) at a specific elevation. If the geographic coordinates of the ground coordinates are requested, the elevation specified is assumed to be the elevation above the ellipsoidal model of the earth. In this case, an iterative algorithm was developed which performs the transition from the "flat earth" concept of the local vertical coordinate system to the geographic coordinates and the curvature of the earth away from the plane within the local vertical system.

- Since no definite experiments were defined, the concept of "photo view frame" of the AFES was not generalized or modified for inclusion in the DDT system. It was decided that such a structure, which is appropriate for AFES, would be awkward, if not also a limitation, within the DDT system.

- An interior orientation parameter computation program was implemented which uses the large image roaming software. A non-display interactive version was also implemented. With this capability, the calibrated sensor coordinates can be selected interactively from any position within the image.

4.3 DIGITAL TERRAIN ELEVATION DATA

The mensuration software of the DDT system is intended to provide a capability to examine the relative accuracies of stereo mensuration, mono mensuration assuming a nominal earth model, and mono mensuration assuming that digital terrain elevation data (DTED) is available over the area of interest. The latter technique requires the intersection of a "projected ray" from the sensor position with the DTED surface. During the course of the DDT A&E contract, another RADC effort was addressing many different terrain intersection algorithms, with the goal of determining the best one. Because the terrain intersection algorithm was not selected yet by the other contract, the terrain intersection algorithm which RRC was directed to implement was the "nearest neighbor". This algorithm is defined to be: given a ground coordinate, extract the elevation coordinate closest to it. Users of this algorithm should be warned of false coordinates because the effects of rough terrain will result in "missed peaks" when the ray is not near the vertical.

Software provided, in addition to the "nearest neighbor" algorithm includes a routine to read a magnetic tape containing the DMA DLMS formatted DTED information. This software will extract a subset of the data contained within the first data file on the magnetic tape. If there are additional data files on the same magnetic tape, the software must be modified because of the "sentinel" files and records associated with the DMA format. Also provided are a pair of programs to convert the disk terrain data header file from binary to ASCII and back again.

4.4 IMAGE COMPRESSION AND DECOMPRESSION

In anticipation of the delivery of the DeAnza IP-8500, which was thought to be similar to the AFES (IP-5500) display systems, the image compression algorithm was implemented to use the IP-5500 series display as well as the AFES structured image files. The IP-5500 provided random access to windows of image data, thus it was anticipated that the information retrieval would be faster than disk accessing. Thus, two pair of programs were implemented. The first pair performs image compression; one using the display as its input source, and the other using a disk file as its source. The second pair performs the image decompression; in this case the image destination is either the display or the disk.

Performing the image compression requires a compression method. Since there was no a priori knowledge how the image data should be quantized, a statistics generating program was required to process the transformed coefficients and arrive at the appropriate quantization scheme.

The result of this portion of the software development is that an image compression/decompression scheme exists on the DDT system. Since the software was developed during the early phase of the contract, the incompatibility of the IP-5500 and the IP-8500 display systems was not known. Likewise, the design for the DDT tessellated image format occurred after the completion of the compression/decompression implementation. Because the relative priorities of the various tasks placed the image compression as very low priority, the software modifications were never made to perform this processing with the IP-8500 or with tessellated images.

5. THE DDT SYSTEM ADMINISTRATIVE CONTROL

A significant amount of time can be, and usually is, lost in any software development project due to insufficient control of programs during the development, integration, and subsequent modification cycle. The greater the number of individuals involved, of course, the greater the impact of poor program management. In the DDT system environment, all source files which are to be incorporated into the DDT system as an include file, library routine, command, etc., are first placed under centralized software control of the DDT System Administrator.

5.1 CENTRALIZED SOFTWARE CONTROL

Software control is the mechanism by which the DDT system accomplishes the task of maintaining continuity in a dynamic programming environment (testbed). The mechanics of software control involve centralized file ownership, monitoring of file editing, and retention of intermediate changes (versions) to files.

5.1.1 File Ownership

With any project where multiple programmers are providing input into the system, be it system or application programming, one of the major obstacles to smooth integration is the problem of multiple versions of routines being used. Invariably someone needs a routine for a specific application. Presuming he hears that someone has written such a program, he often cannot locate the current version. Then if he needs to modify it even slightly he usually creates a new file. The tendency here is for a proliferation of special purpose programs. This does not encourage the programmer to work in a modular environment. If many people have incorporated a routine into their routines then the task of update in case of change becomes enormous. One of the major requirements for system development to proceed at any kind of reasonable pace in the above scenario is an immense overall knowledge of the system by a few individuals. If for some reason these people leave the project, it may require months for the project to recover.

Centralized file ownership is one of the ways the DDT system avoids the above problems. All files which make up the DDT system are owned by the DDT system administrator. They are stored in one of the DDT SCCS directories according to their suffix. All program development commands access these files in various ways for the programmer without requiring him to know where the actual source files are located. He can therefore be sure he has the correct copy of the file. He also has many commands at his disposal to determine what routines are available in the DDT system and gather information about them. He is encouraged to write routines which may be beneficial to others through provision of an easy-to-use interface to the DDT libraries. Easy access to all DDT files also allows a programmer to explore the software to any depth desired. The user may place a file under DDT SCCS control via "addfile", "add_to_ddt". The "addfile" command merely places the source file under the DDT system centralized control in the form of an SCCS file. The other command will execute "addfile" as required.

5.1.2 Source Code Control System (SCCS)

Having centralized ownership of files, the next step is to control the modification of files. This task is accomplished using the PWB product called the Source Code Control System (SCCS). SCCS stores the original version of a file and all subsequent modifications to it. Any version of the file can be produced by applying the modifications or "deltas" (as they are referred to by SCCS), up to the the version desired, to the original file.

5.1.3 File Editing

When editing a file under DDT system control, the user always gets the latest version. If he wishes some previous version to be the latest he may execute the backup command. He never loses any of the versions, however. One difference the programmer will notice between the listing of a file he has retrieved via editfile from the same version retrieved via the catfile command will be the absence of any date or time information in the documentation boilerplate. Instead, the programmer will notice some capital letters which are preceeded and followed by the % character. These are recognized by SCCS and the appropriate substitutions for them are made when a listing of the program is requested. They provide the programmer information as to the version number, the date of the listing, date of last update, etc.

5.2 SYSTEM UPDATE

After a programmer has made changes to a file which is under DDT system control he needs to be sure that the changes are reflected throughout the system. This is accomplished by the DDT system administrator. A file in DDT which is intended to be used in the system, be it an include file, subroutine, or main routine, must be placed under DDT Make Control. The command to do this is add to ddt.

5.2.1 File dependencies

The concept of file dependency means that one entity in the system is dependent on one or more files in the system. Whenever any of these other files is modified the entity needs to be updated in some manner. The entity may be no more than a file which contains the contents of two other files; in case of a change to either of the files, the entity is reloaded with current versions of the files. The entity may be, however, a complex executable module which is dependent on a number of include files, library subroutines, and other source files. If any of these files change the module must be recompiled and loaded.

5.2.2 Make command

The Make command in UNIX/PWB provides a mechanism by which the system can be kept up to date in a semi-automatic manner. The DDT system administrator is the only one who actually executes this command and he does so indirectly via the ddtupdate or tstbed make(tst). These routines move around in the DDT directories and get an updated copy of the Makefile for the directory and then execute the Make command. The following is an excerpt from the makefile for the "obj1" directory as an example:

```
#       Makefile
#       will make all the feature commands in /u/ddt/obj$z
#       This is a release dependent makefile and all programs
#       are dependent upon shell variable $z for release #.

CC =  /bin/cc -O -DPWB
FC = /bin/cc -O -I2
LIB = /u/ddt/lib$z
INCL = /u/ddt/incl$z
FILES = /u/ddt/sccs/files
OBJ = /u/ddt/obj$z
PROG = /u/ddt/cmd
MODUL = /u/ddt/bin$z/modules

update : makeall

makeall :: mod_fr.o
mod_fr.o : $(FILES)/s.mod_fr.c $(INCL)/photo.h
        $(PROG)/copyfile mod_fr.c
        $(CC) -c mod_fr.c -I$(INCL)
        -rm -f mod_fr.c

makeall :: prt_hdr.o
prt_hdr.o : $(FILES)/s.prt_hdr.c $(INCL)/ddt_im_hdr.h \
        $(LIB)/error.o $(LIB)/findargs.o \
        $(LIB)/df_loc.o $(LIB)/df_r_hdr.o
        $(PROG)/copyfile prt_hdr.c
        $(CC) -c prt_hdr.c -I$(INCL)
```

```
                -rm -f prt_hdr.c
```

The purpose of this excerpt from the object Makefile is to keep the object code up to date for mod_fr.o and prt_hdr.o. After the object directory has been brought up to date, the Makefile in bin$z will load the object module with appropriate subroutines and libraries. The two are maintained separately for clarity. The lines at the beginning of the excerpt are comments as are any lines preceeded by the "#" sign. Next are a list of macro definitions which may be substituted in the body of the Makefile with the $(NAME) string, where NAME is the string to the left of the "=" in the macro definition. The first executable line of the Makefile is always made if no argument is given, so it is a convention of DDT to have a dummy line there which is dependent on "makeall" which is dependent on all the items defined in the Makefile. When "make" is executed, all items are checked and updated as required. The single colon indicates dependency and the double colon allows for a continuation of dependencies. In the case of mod_fr.o, it is dependent on the SCCS source file for mod_fr.c, and the include file photo.h. If either of these two files changed since the last time mod_fr.o was created, then the make command would execute all of the lines following the line of dependencies up to the next item. In this case it will get the current copy of mod_fr.c, compile it and then remove the source file mod_fr.c from the object to further prevent any proliferation of uncontrolled source files. In the case of prt_hdr.o, it is dependent on the SCCS source file for prt_hdr.c, the include file ddt_im_hdr.h, the ddt$z library subroutines error.o, findargs.o, df_loc.o, and df_r_hdr.o. The make will proceed as with mod_fr.o.

5.2.3 Placing a Command Under DDT Make Control

As mentioned earlier, the command which allows one to place an item under DDT Make Control is "add_to_ddt". This complex command is dependent on the highly structured DDT directory layout. This command allows one to start with either a high level item such as a command or at a lower level such as an include file. If one starts with a command, the SCCS files will be searched for the main routine or shell file as appropriate. If it is not found, the routine "addfile" will be executed automatically for the routine. The user will then be asked questions such as file dependencies and the same process will be repeated for all lower level routines. Finally the Makefiles associated with the command will be modified automatically and the user will be prompted to add appropriate help and menu entries.

6. CONCLUSIONS AND RECOMMENDATIONS

The software developed for the DDT system in the EPF provided the following features:

1. Modular software design which will permit the inclusion of an arbitrary number of new or different sensor models.

2. Basic framework for experimenting with multiple sensor models for an image.

3. Modification to the SCM software to permit monoscopic point selection and ground coordinate derivation.

4. Large image roaming with zooming and coordinate capture capability.

5. Split screen viewing with large image roaming, zooming and coordinate capture capability.

6. Mathematical transformations to compute image coordinates from display coordinates, sensor coordinates from digital image coordinates and ground coordinates from sensor coordinates.

7. Modularization of the SCM and AFES mensuration code to provide a flexible framework to perform experimentation.

8. Contiguous file system and high speed file routines for image handling optimization for the large image roaming capability.

9. On-line documentation of the DDT system software.

10. Administrative control over the DDT system software source, documentation and executables.

The major deficiencies of the DDT system are the following.

Stereo viewing: The DDT system did not provide an integrated stereo viewing capability. The absense consist primarily of no operator interface between viewing two images in a "superimposed" fashion and selecting display coordinates from both images. The images would be stereo pairs displayed in the red and green channels permitting anaglyph viewing. Software already exists on the RADC IPS using the IP-5532 display for images having a size of 512x512 and having the appropriate relative orientation to each other.

Incomplete image warping code: It is essential to have an image warping capability to perform the required relative orientation to align the epipolar lines of a pair of images for stereo viewing. Warping software was written using an affine transform and using the IP-8500 display system as the image input and output device. The software was not tested on the DDT system and thus, never placed under DDT adminsitrative control.

As in the case of stereo viewing, software already exists on the RADC IPS to warp an image from one IP-5532 image display channel to that of another using various warping transformation.

Warping transformation parameter generation: The DDT system does not have an operator interface to interactively generate a set of warping parameters to be used to perform an image warp. The major display code required to perform this operation is contained in the large multi-image roaming program "maintb" and the interior orientation parameter computation found in the programs "ddt_int_or" and "s_int_or".

Image enhancements: Time did not allow the implementation of image enhancements routines that already existed on the RADC IPS using the IP-5532 display. A menu containing the RADC IPS abilities is included within the DDT system. This menu is to serve as a "pointer" to the program names on RADC IPS so that the source files may be obtained and modified as desired.

Point transfer capability: The DDT system does not have a user interface to perform a computer assisted transfer of image position from one image to a corresponding position on another image. The method to perform this operation would be similar to that of generating the image warping transformation parmeters. The basic starting point for implementing such a program is a combination of the program "maintb" and "ddt_int_or".

The recommendations for the improvement of the DDT system include, besides the obvious elimination of the deficiencies cited above, are as follows:

1. Include the panoramic sensor model in the sensor geometries (software for this already exists on the DMA versions of the RADC IPS).

2. Include an epipolar model method of computing the image warping coefficients to enable stereo image viewing. This software exists for the DMA versions of the RADC IPS and is integrated into the image scanning parameters for use on the scanners at the DMA Centers.

3. Expand the frame photo header to provide for other coordinate systems besides geographic coordinates as the origin of the local vertical system. UTM coordinates should be included in any redesign of the header.

4. Integrate the appropriate DTED intersection program with monoscopic point positioning program. That is, the DTED intersection algorithm must be included within the sensor to ground projection, otherwise, false terrain intersections could be derived as a result of irregular terrain.

DATE

ILME